ground water

Rapid Communication/

Using a Cloud to Replenish Parched Groundwater Modeling Efforts

by Randall J. Hunt[1], Joseph Luchette[2], Willem A. Schreuder[3], James O. Rumbaugh[4], John Doherty[5,6], Matthew J. Tonkin[7], and Douglas B. Rumbaugh[4]

Abstract

Groundwater models can be improved by introduction of additional parameter flexibility and simultaneous use of soft-knowledge. However, these sophisticated approaches have high computational requirements. Cloud computing provides unprecedented access to computing power via the Internet to facilitate the use of these techniques. A modeler can create, launch, and terminate "virtual" computers as needed, paying by the hour, and save machine images for future use. Such cost-effective and flexible computing power empowers groundwater modelers to routinely perform model calibration and uncertainty analysis in ways not previously possible.

Introduction

Societal decision-making on water issues—both quantity and quality—requires science-based tools such as computer models. Numerical models are vital for informing such decisions because the models can be used to investigate a range of actions using a quantitative and physically based framework, which in turn facilitates reactive as well as proactive action. However, groundwater models can have long runtimes; large, transient groundwater flow, contaminant transport, and coupled groundwater-surface water models can take more than a day to complete a single run. Such long runtimes are an impediment to investigating alternate conceptual models and alternate management scenarios with models—especially when performing model calibration and sensitivity analysis where the model is run many times.

At the November 2009 PEST (Parameter Estimation) Conference in Potomac, Maryland, the recent development of "cloud computing" was discussed as a means to bring unprecedented computing power to bear on groundwater problems (Luchette et al. 2009; Schreuder 2009). Cloud computing has been widely covered in the recent popular press (e.g., http://www.newsweek.com/id/140864), and in its simplest form includes Internet-accessible e-mail. However, cloud computing also includes other capabilities, including allowing customers to create multiprocessor configurations, or "supercomputers," by renting virtual computers over the Internet. Thus, cloud computing allows the modeler to access the number of machines that best suit the modeling problem rather than restricting that number to those machines available locally. In a common parallel computing application, multiple processors are used to reduce runtimes by parallelizing a single model run. For example, Arnett and Greenwade (2000) reported an almost fivefold reduction in runtime when a contaminant transport model run on a single processor was split among 10 parallel processors.

Many groundwater models are not well suited for parallel computing, however, because communication

[1]Corresponding author: U.S. Geological Survey, 8505 Research Way, Middleton, WI 53562; (608) 821-3847; fax: (608) 821-3817; rjhunt@usgs.gov

[2]McLane Environmental LLC., Princeton, NJ.

[3]Principia Mathematica, Lakewood, CO.

[4]Environmental Simulations Inc., Reinholds, PA.

[5]Watermark Numerical Computing, Brisbane, Australia.

[6]National Centre for Groundwater Research and Training, Flinders University, Adelaide SA, Australia.

[7]S.S. Papadopulos & Assoc. Inc., Bethesda, MD.

Received January 2010, accepted February 2010.

doi: 10.1111/j.1745-6584.2010.00699.x

overhead between processors can offset the gain of adding processors. Therefore, such speedups cannot be universally expected even with additional computing capability provided by cloud computing. However, all models can benefit from another parallel computing application—automated calibration and uncertainty analysis using parameter estimation techniques. During the parameter estimation process, each user-specified model parameter is adjusted and the model outputs are compared with corresponding field measurements. The effect of each parameter change on the model-generated counterparts to field observations is used to develop an updated estimate of the optimal parameters. Hence, a large number of runs must be performed to compute the updated parameter set that improves a model's fit to the data. Fortunately, parameter estimation has properties of an "embarrassingly parallel" problem (Foster 1995), thus making it well suited for parallel (and thus cloud) computing. Three aspects make this the case: (1) the runs are completely independent of each other (no interaction between runs); (2) all the runs can be decided before any run is launched; and (3) the runs are idempotent, that is, doing the same run more than once has no side effects. Given this universal application to groundwater models, indeed to all environmental models, the remainder of the discussion will focus on the application of cloud computing to parameter estimation problems.

Parameter estimation, like all groundwater modeling, confronts a common problem—the natural world always has more complexity than can be included in any model parameter set. To the extent that processes and characteristics are so simplified that observed system behavior is not completely replicated in a model, so-called "structural noise" (e.g., Doherty and Welter 2010) degrades model outputs that correspond to these observations (as well as predictions that the model is required to make). The more salient information that is omitted, the more *over*-simplified the model and the larger its structural error. These deficiencies in model behavior can be addressed to some degree through use of appropriate complexity, attained by using higher numbers of parameters than have been traditionally included. This increased model flexibility can help reduce structural noise by allowing model parameterization to be more receptive to information contained in calibration data, which in turn can reduce the potential for error in model predictions. Moreover, new complimentary methods have taken advantage of insight gained from highly parameterized models to better estimate the potential for prediction error (e.g., Moore and Doherty 2005).

Why not run all models with hundreds or thousands of parameters as a means to maximize model flexibility and keep the structural error associated with omitted detail small? Estimation of parameters for overly complex models can be unstable and nonunique, though these problems can be overcome with a "regularized inversion" approach where large numbers of parameters are constrained using mathematical methods and soft-knowledge of the system (Hunt et al. 2007). Carrying many parameters during model calibration, however, still carries high computational costs; most parameter estimation methods commonly require at least one model run per parameter during each iteration. Even mathematical enhancements such as the use of "super parameters," (linear combinations of base parameters, Tonkin and Doherty 2005), require an initial sensitivity analysis where sensitivity of all model outputs to each parameter must be calculated before defining the more limited number of super parameters. Although such enhanced methods are now routinely used in everyday modeling practice, the upper limit on the number of parameters is often still chosen based on the number of computers available, and not on what is best suited for calibration, or for analyzing the uncertainty of a prediction of interest. Thus, these artificial and arbitrary constraints to model parameter estimation and uncertainty analysis could limit our ability to bring the best science to water resources decision-making. Cloud computing is a powerful new tool that can overcome this restriction (Luchette et al. 2009).

Cloud Computing

The term "cloud computing" refers to leveraging the computing capability of others through the Internet as a service. Distant computing power resides in "virtual machines"—virtual computer resources that mimic actual machines running a single process or entire operating systems. The types of services offered using cloud computing usually fall into one of three categories: (1) software-as-a-service (SaaS); (2) infrastructure-as-a-service (IaaS); and (3) platform-as-a-service (Figure 1). SaaS makes otherwise traditional software available as an online utility. Webmail (e.g., Gmail, Yahoo) is probably the most common and simplest example of SaaS: e-mail is stored on distant servers but accessed by users through a local web browser. The third, platform-as-a-service, consists of a sophisticated and encompassing framework that allows applications to be built in one or more specific programming languages and rolled-out on the service provider's computing resources while scaling automatically to increased user demand. Between these end members is IaaS, the area of cloud computing likely to have most utility to environmental modeling. IaaS provides users instant access to virtual machines that contain entire operating systems via a web-based interface or through application programming interfaces (APIs). Such capability enables vast multiprocessor computing capabilities to be built without large capital expenses of hardware and software. Any number of virtual machines can be brought to bear on a problem; thus, IaaS allows computational power to scale efficiently to the modeling need.

A cloud service rigidly offering only one code as a SaaS would result in an unsatisfactory "one size fits all" approach. Yet building environmental models does not require the integrated sophistication (and associated expense) of a platform-as-a-service application. IaaS is attractive in that it possesses easily accessed

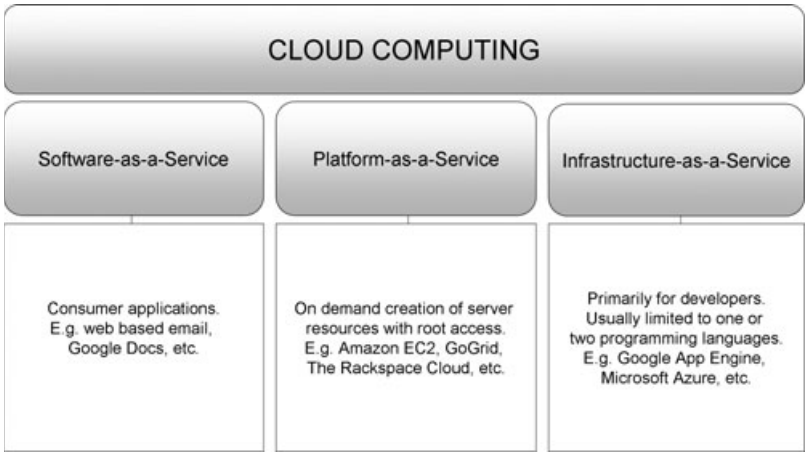

Figure 1. Examples of cloud computing and services offered, with brief explanations of each service (modified from Luchette et al. 2009).

web-based interfaces for starting and stopping virtual machines, this allowing the user to easily acquire and terminate the required computing resources. Furthermore, the customer is granted full "root access" to a created virtual machine. Thus, a modeler can upload, setup, and run software applications as on a local desktop machine. Private networks between virtual machines can be constructed via folder sharing or Internet protocols on the cloud—thereby setting up a scaleable parallel computing environment for model calibration tools such as Parallel PEST (Doherty 2010). Once a virtual machine is configured, it can be saved for future use, reducing the effort needed to start subsequent cloud sessions. Finally, IaaS resources are becoming more widespread: examples include Amazon EC2 (http://aws.amazon.com/ec2), GoGrid (http://www.gogrid.com), and The Rackspace Cloud (http://www.rackspacecloud.com).

Perhaps the most appealing feature of IaaS cloud services is the inexpensive metered pricing. Similar to a household utility meter, costs are calculated on the basis of the amount of time cloud resources are used. This allows users to allocate large amounts of computing resources for set lengths of time, without incurring the upfront capital expense of purchasing the equipment. Pricing models are typically based on virtual machine memory and/or processor speed and are billed by the hour. The cost currently is typically in the range of $0.08 to $1.20 per hour per processor depending on the memory and processor specified. Additional pricing for data transfer is also applied but is typically low (approximately $0.10 to $0.50 per gigabyte [GB] per month).

Harnessing the Cloud

Acquiring computing power solves only one aspect of the problem; the modeler must also efficiently utilize the resources provided by the newly available tool. Schreuder (2009) outlines a parallel parameter estimation approach suitable for implementation on the cloud using an extension of Parallel PEST. PEST is the most widely used parameter estimation code for groundwater modeling (Ginn et al. 2007) and has been extensively run on local parallel computing networks. PEST for one processor is expanded to "Parallel PEST" on multiple processors where a single supervisor (or "master") distributes model runs to individual computers across a network. The master creates the model input files in a designated directory, instructs the slave to launch the model, and then reads the results from the same directory upon completion of model execution. Communication between the master and slaves is effected through small message files; this approach works well for a smaller number of slaves but does not scale well to high numbers of slaves. When there are many slaves, the master becomes the bottleneck because it becomes bogged down with writing the large number of required input files and reading the output files that each model instance generates, files that are often large. Because the master writes the files, it is also necessary to have a global file system visible to the master and all slaves. This is clearly not well suited for integrating local computing resources with virtual machines accessed through the cloud.

Recently, Parallel PEST was refined to improve parallel performance through development of BeoPEST (Schreuder 2009). BeoPEST implements the same parameter estimation algorithms as Parallel PEST but supports two other communications protocols in addition to the message file approach. BeoPEST/MPI uses the message passing interface (MPI) protocol to communicate between the master and slaves and is ideally suited to use on supercomputers—the "Big Iron" of computing. BeoPEST/TCP uses the Internet standard Transmission Control Protocol/Internet Protocol (TCP/IP) protocol to communicate between the master and slaves; this facilitates cloud-based parameter estimation because it allows the master and slaves to run on any computers that can communicate via the Internet. It is also well suited to ad hoc clusters that can be formed using office personal computers (PCs), either locally or in distributed offices connected by the Internet.

Enhanced parallel computing also requires a smarter slave approach than that originally used by Parallel PEST. With such an approach, the master simply sends a short message to the slaves indicating the values of parameters that the model must use during the next model run. The slaves takes care of writing the model input files using local templates of these files, running the model, and extracting the results into a compact format. Writing the input files and reading the output files may seem trivial; however, having the slaves perform this task locally provides three significant benefits. (1) Performance: if it takes just 1 s per client to read and write the files, it would take a single server 1 h to perform the same reading/writing tasks as 3600 smart slaves can accomplish in a second. (2) Locality: all the files can be written to the local disk connected to the slave; therefore, a global file system is not required. (3) Reliability: because TCP/IP is a reliable protocol, the master can sense when a slave dies or can no longer be contacted. If a slave fails, BeoPEST simply reschedules the run that was being executed by that slave.

BeoPEST allows a modeler to take advantage of both local desktop and cloud resources (Figure 2); thus, cloud resources can be used to augment local computing capabilities on-demand. The master can run on the user's desktop, whereas the slaves can be leased from a cloud provider in addition to running on machines in local and satellite offices. The user simply copies the necessary files to the cloud file system and satellite offices, starts the master on a desktop computer, and then starts the cloud and satellite office slaves while providing them with the IP address of the desktop system. Although it is possible to run both the master and slaves in the cloud, it is advantageous for the modeler to monitor the progress of a large optimization exercise; this is better performed on a local desktop. BeoPEST/TCP also supports heterogeneous environments where the slaves run on different operating systems (e.g., Linux, OSX, and Windows) or even different hardware (e.g. SPARC or POWER) than the master.

Computer and network security is an increasing concern, and these concerns extend to accessing the cloud. BeoPEST handles security by sending only numerical values of parameters and observations across the Internet. These data are then used to generate numerical values for input files by a smart slave. The standard user-supplied PEST control file selects which programs to run as the model. Maliciously corrupting the master-slave communications will cause the parameter estimation process to go awry but cannot cause slaves to execute arbitrary or malicious code. Moreover, because cloud computing allows users to save known machine images, new virtual machines always start at a known configuration.

A Cloud Example Using Parallel PEST

With the variety of cloud vendors and BeoPEST protocols, more permutations can be implemented than can be covered here. A simple test of the speed of cloud computing virtual machines is demonstrated using the GoGrid IaaS cloud system. Because cloud computers work just like a local computer or laptop, model runs require uploading of files and model executables to the cloud server. In this example, this was accomplished

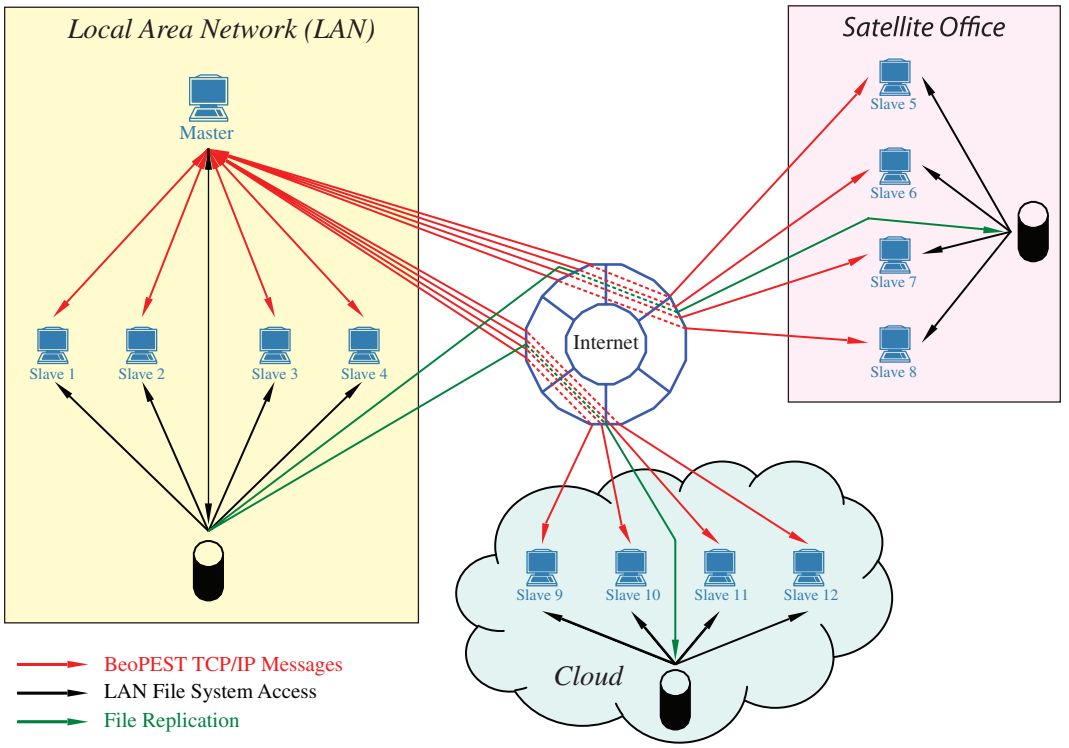

Figure 2. Schematic of parallel computing on the cloud using BeoPEST (modified from Schreuder 2009).

through the Windows remote desktop protocol (RDP). A single virtual machine can then be cloned as many times as required. However, the IP address is usually not static as in a local computing environment. As a result, one current limitation of cloud computing is the lack of robust scripts/batch programs that automatically clone, load, and launch slaves while accounting for the dynamic nature of computing resources on the cloud.

A regional groundwater flow model described by Luchette et al. (2009) was used to compare the speed of a virtual machine on the cloud to that of typical local computers. The model consisted of a 5-layer, 50-stress period MODFLOW2000 (Harbaugh et al. 2000) simulation. The bottom of the model was specified as no-flow; the top of the model received recharge. Constant flux boundaries were specified at the model perimeter and a river was represented using a series of river cells bisecting the model domain in layer 1. As expected given the GoGrid virtual machine minimum configuration, model runtimes obtained from the virtual machine were comparable with those obtained on local desktops (Table 1).

The power of cloud computing is more evident during parameter estimation. Luchette et al. (2009) describe a 28-parameter problem that required 1691 runs for parameter estimation completion. This problem would require between 26 and 41 h on a single processor that had runtimes similar to those reported in Table 1. Luchette et al. (2009) ran this problem as a Parallel PEST run on a four-slave configuration on the GoGrid cloud in approximately 9 h. A comparable run time was also obtained when running four slaves on a dedicated local modeling array. When run on 28 slaves on the cloud, the total runtime was just over 2 h (Figure 3). MODFLOW2000 runtimes were consistently near that of the single forward run, with some differences noted between the virtual machine employed (Supporting Information Table S1).

This modest test of the cloud is consistent with a common parameter estimation practice of using as many slaves as estimated parameters and illustrates the cloud's ability to consistently deliver runtimes comparable with local desktop computers. Moreover, the ability to scale computational power to the needs of a given problem is important because the number of slaves is often the most important factor for determining the total time needed to perform parameter estimation on a given conceptual model. If one parameter estimation run is considered to constitute an evaluation of a particular conceptual model, then the increased capability gained from cloud computing in this example represents an ability to evaluate three to four conceptual models a day compared with one (four slaves) or less (one nonparallel processor) per 8-h workday. This represents an appreciable gain in the exploration of alternative model conceptualizations and in the quality of overall model insight gained through calibration. Furthermore, the concept of "low cost" has new meaning in a cloud computing world: the total cost of the 9-h simulation that involved 1691 calibration runs using four slaves on the cloud was less than $5.00 (Luchette et al. 2009). The cost using a 2-h, 28-slave run on a cloud array was less than $15.00—much lower than even a single, low-end, stand-alone computer purchased for a local network.

Table 1
Comparison of Runtimes of One Model Run

Computer	Time (s)
Q6700 Core 2 Quad (2.66 GHz)	85
GoGrid Cloud virtual machine	81
Xeon (3.0 GHz)	73
Q9650 Core 2 Quad (3.0 GHz)	71
i7 (3.33 GHz)	58

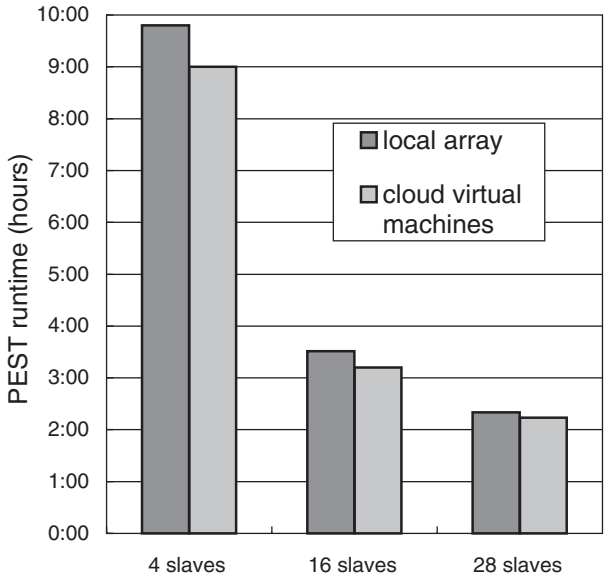

Figure 3. Comparison of parameter estimation runtimes obtained from a dedicated local desktop array and virtual machines run on the cloud.

Summary

Groundwater models can be improved by parameter flexibility together with simultaneous use of soft-knowledge to constrain the increased number of parameters. However, parameter estimation techniques, especially in a highly parameterized context, can still have high computational costs. Cloud computing allows modelers to use the Internet to rent virtual computers to process their data or run their own computer applications. It allows on-demand access to virtual computing resources whereby a modeler can create, launch, and terminate virtual machines as needed, paying by the hour for active processors, and saving these "machine images" for future use. Such cost-effective and flexible computing power allows groundwater modelers to approach model calibration and uncertainty analysis in ways not previously possible.

Supporting Information

Additional Supporting Information may be found in the online version of this article:

Table S1. Cloud runtimes from 2-h PEST parameter iteration run.

Please note: Wiley-Blackwell is not responsible for the content or functionality of any supporting information supplied by the authors. Any queries (other than missing material) should be directed to the corresponding author for the article.

References

Arnett, R.C., and L.E. Greenwade. 2000. Parallel processing of a groundwater contaminant code. In *Conference Proceedings of Summit 2000, the 42nd Annual Cray Users Group Meeting*, Noordwijk, The Netherlands, 22–26 May. http://www.inl.gov/technicalpublications/Documents/2690197.pdf (accessed December 7, 2009).

Doherty, J. 2010. *PEST: Model-Independent Parameter Estimation*. Australia: Watermark Numerical Computing. http://www.pesthomepage.org (accessed December 7, 2009).

Doherty, J., and D.E. Welter. 2010. A short exploration of structural noise. *Water Resources Research*. In press. DOI: 10.1029/2009WR008377.

Foster, I. 1995. *Designing and Building Parallel Programs*. Upper Saddle River, New Jersey: Addison-Wesley Pearson Education. ISBN 9780201575941, 430 p.

Ginn, T.R., T.D. Scheibe, H. Haeri, and C.N. McClain. 2007. Paper 40–1: an expanded survey of groundwater modeling practitioners about how they quantify uncertainty: which tools they use, why, and why not. *Geological Society of America Abstracts with Programs* 39, no. 6: 40–41.

Harbaugh, A.W., E.R. Banta, M.C. Hill, and M.G. McDonald. 2000. MODFLOW-2000, the U.S. Geological Survey modular ground-water model—user guide to modularization concepts and the ground-water flow process. U.S. Geological Survey Open-File Report 00-92. Reston, Virginia: USGS. 121 p.

Hunt, R.J., J. Doherty, and M.J. Tonkin. 2007. Are models too simple? Arguments for increased parameterization. *Ground Water* 45, no. 3, 254–262.

Luchette, J., G.K. Nelson, C.F. McLane, and L.I. Cecan. 2009. Unlimited virtual computing capacity using the cloud for automated parameter estimation. In *Proceedings of the 1st PEST Conference*, Potomac, Maryland, 1–3 November.

Moore, C., and J. Doherty. 2005. Role of the calibration process in reducing model predictive error. *Water Resources Research* 41, W05020, DOI: 10.1029/2004WR003501.

Schreuder, W.A. 2009. Running BeoPEST. In *Proceedings of the 1st PEST Conference*, Potomac, Maryland, 1–3 November.

Tonkin, M.J., and J. Doherty. 2005. A hybrid regularized inversion methodology for highly parameterized environmental models. *Water Resources Research* 41, W10412, DOI: 10.1029/2005WR003995.

Author's Note: The use of brand, trade, or firm names in peer-reviewed papers is for identification purposes only and does not constitute endorsement by the authors, their employers, or the National Ground Water Association.

Supplemental Material Table S1. Cloud runtimes from 2-hour PEST parameter estimation run

Virtual Machine#	Slave	No. Runs	Minimum time (sec)	Average time (sec)	Max time (sec)
1	slave1	57	80	84	105
1	slave2	54	79	84	107
1	slave3	56	79	84	108
1	slave4	56	77	85	109
1	slave5	54	80	86	110
2	slave6	76	72	80	107
2	slave7	78	75	79	92
2	slave8	79	74	79	93
2	slave9	79	74	80	90
2	slave10	75	75	82	93
2	slave11	74	75	82	94
3	slave12	63	73	84	95
3	slave13	66	74	83	96
3	slave14	62	74	85	97
3	slave15	61	76	85	98
3	slave16	62	75	85	99
4	slave17	53	83	90	104
4	slave18	53	83	90	103
4	slave19	53	83	91	101
4	slave20	53	83	92	102
4	slave21	53	84	93	108
4	slave22	53	83	94	104
5	slave23	55	76	93	109
5	slave24	53	79	94	110
5	slave25	55	79	94	111
5	slave26	53	80	96	113
5	slave27	53	80	97	115
5	slave28	53	80	97	117

From:

Hunt, R.J., Luchette, J., Schreuder, W.A., Rumbaugh, J.O., Doherty, J., Tonkin, M.J., and Rumbaugh, D.B., 2010. Rapid Communication: Using a Cloud to replenish parched groundwater modeling efforts. *Ground Water*, doi: 10.1111/j.1745-6584.2010.00699.x